# AMYGDALA $z \mapsto z^2+c$

**A Newsletter of Fractals & $\mathcal{M}$ (the Mandelbrot Set)**

Issue #30
March 1993




## In This Issue...

Michael Ackerman's *A New Look At Koch Curves* discusses the family of "linear" fractals of which the Koch Curve (Snowflake) is the primordial example.

## The Slides (S30)

Only about a third of you subscribe to the slide supplement, and several of you have complained about having the issue occupied with a column that they're not interested in, since they don't get the slides.

Starting with this issue I'm going to honor that by sending the slide set to those who subscribe for it, accompanied by a separately printed description.

## A New Look At Koch Curves

*— Michael Ackerman*

Benoit Mandelbrot devotes much of *The Fractal Geometry of Nature*[1] to *Koch curves*: variants on a jagged shape devised by Helge von Koch. These figures are defined by their method of construction: an open curve consisting of several line segments is copied and scaled to replace each of its components. Mandelbrot's intricate images result from limited repetition of this procedure, rather than the infinite repetition that would produce theoretical limit curves.

It is somewhat frustrating that many "monster" curves cannot be created by a Koch construction. Mandelbrot shows how Hilbert's well-known curve can be built using a Koch curve as a guide, but provides no direct Koch construction. Giuseppe Peano's original curve *can* be produced with the Koch method. Unfortunately the curve is self-contacting, so it ends up looking like graph paper. The traditional remedy is to round off or clip the corners. "Aleph-Null"[2] took this notion to an extreme with a curve that connects the mid-points of adjacent segments of Peano's curve, effectively clipping off so much of the corners that they disappear! The results of these modifications are not Koch curves, however.

String-rewriting systems or L-systems can draw curves that the Koch method can't. Such systems are actually extensions of Koch's method. In addition to line segments, these systems can replace the turns and spaces between segments. They also allow the insertion of graphic elements into a curve.

I took a different approach in drawing these shapes. Instead of using a formal system of symbols, I wrote out working Macintosh QuickBASIC programs. The reason for this is that I was unable

1. See reference [Man82].

2. See reference $\aleph_0$.

**CONTENTS**

to reproduce some figures using L-system interpreters described in popular publications. This may have come from my own ignorance, but I think that the L-system I used was just too limited. I hope that these programs are an acceptable alternative to a home-brew interpreter. I've tried to make them readable, with whole lines corresponding single turtle-graphics statements.

## First Things First: The Koch Snowflake

All discussions of self-similar curves seem to include von Koch's snowflake, but it is useful to take another short look, as it sets the pattern for the programs that follow. The program of listing 1 draws a Koch arc by the usual method: each segment is replaced by a miniature version of the curve as a whole. The first part of the program sets up variables and makes one call to the program's second part.

The second half of the program is the interesting part. It consists of a subroutine which calls itself, using the variable rder to track its level of recursion. An IF...THEN...ELSE statement decides whether to draw a line segment, or to go further in recursion. The "recursion" side of the conditional simply alternates GOSUB statements with turn commands.

The result is shown in Fig 1.

The program of listing 2 draws the Koch snowflake in a subtly different way. The conditional statement in the subroutine does not choose between two sets of statements; it only decides whether or not to use one group of GOSUB, turn, and line-drawing statements. The resulting snowflake comes out with the final segment missing, so a line statement is added to the initial part of the program. Drawing a Koch curve without the Koch method might seem pointless, but it illustrates a way to draw non-Koch curves.

## Peano Variations and Space Dragons

The program of listing 3 draws Peano's grid-like monster without the Koch method. Like listing 2, it contains a single subroutine, a "one sided" conditional (IF and THEN, but no ELSE), and a tacked-on final segment. The program of listing 4 draws a curve which connects the midpoints of adjacent segments in the Peano curve. The surprising thing is that these two programs are almost identical. One can be converted to the other simply by swapping a few lines. The tacked-on line can even be eliminated in the self-avoiding curve.

The programs of listings 5 and 6 bear a similar relationship. 5 is, of course, the Harter-Heighway dragon curve, with a graph-paper interior. Unlike previous examples, the generating program requires two subroutines, but their structure should be familiar. I've nicknamed the curve from listing 6 the *space dragon* because its interior is more open.

For completeness, here is a shorthand description of an algorithm to draw a dragon curve using Koch's method:

A: -45,A,+90,B,-45
B: +45,A,-90,B,+45

Choose either statement and recursively substitute statements for corresponding letters several times. Then substitute an "F" for each "A" and "B".

This algorithm will produce a string resembling one from an L-system which can control a virtual turtle, with the numbers representing turn angles and the "F"s representing line segments.

## Ordered Dither

The ordered dither [FvD82] differs from the other material in this article in one important way: it has a practical use. It is commonly used to simulate shades of gray on a bi-level display device. I've used it in a program that draws the Mandelbrot set coarsely at first, then gradually in more detail.

Dither matrices are not curves; they define sets of disconnected points.

$$\begin{bmatrix}0\end{bmatrix}, \begin{bmatrix}0 & 2\\ 3 & 1\end{bmatrix}, \begin{bmatrix}0 & 8 & 2 & 10\\ 12 & 4 & 14 & 6\\ 3 & 11 & 1 & 9\\ 15 & 7 & 13 & 5\end{bmatrix}, \begin{bmatrix}0 & 32 & 8 & 40 & 2 & 34 & 10 & 42\\ 48 & 16 & 56 & 24 & 50 & 8 & 58 & 26\\ 12 & 44 & 4 & 36 & 14 & 46 & 6 & 38\\ 60 & 28 & 52 & 20 & 62 & 30 & 54 & 22\\ 3 & 35 & 11 & 43 & 1 & 33 & 9 & 41\\ 51 & 19 & 59 & 27 & 49 & 17 & 57 & 25\\ 15 & 47 & 7 & 39 & 13 & 45 & 5 & 37\\ 63 & 31 & 55 & 23 & 61 & 29 & 53 & 21\end{bmatrix}$$

A program to generate these matrices looks like a Koch curve generator turned inside out. The program of listing 7 writes a dither matrix to the screen

in numerical order. To draw a square graphically, change the three lines in the "setpoint" routine to "PSET (i,j)". The dither matrix of level 3 looks like this:

Incidentally, the anthology *Graphics Gems* [Gla90] contains an algorithm by Jim Blandy and Stephen Hawley which fills a dither matrix by an entirely different technique. It computes elements by manipulating the bits in the variables for the X and Y coordinates and the matrix size.

## Penrose Tiles

The program of listing 8 is clearly different from the preceding programs, and its the only one I'm not entirely happy with. A Penrose tiling looks like a Koch curve, but it cannot be drawn as a continuous non-overlapping line (consider the meeting of three segments). The program I wrote has characteristics of Koch generators and the ordered dither program; it draws line segments, but it also jumps around. If may be possible to draw Penrose tile segments more or less contiguously, but the program would have many more than four routines.

Figure 5 is derived from an "inflation diagram" in Gardner's article [Gar89], and it explains the workings of listing 8. The thin lines indicate where scaled copies are placed, and the bold lines from the lowest level of triangles constitute the Penrose tiling.

## A Note on Program Style

I've written each program to be as simple as possible. As a consequence, the programs do the same things in different ways. With the Koch snowflake it was convenient to perform trigonometric calculations inside the LINE -STEP statements. The Penrose tile program required much more accurate placement of line segments, so its double-precision calculations are performed separately. The Peano and dragon curves make only right-angle turns, so I used a simple technique that swaps *X* values for *Y*. For clarity, an R or L appears at the end of these lines to indicate the direction of turn.

## Notes


[Ack86] Michael Ackerman. "Hilbert Curves Made Simple", *Byte*, June 1986, pp. 137-138. Draws Hilbert's curve with the techniques used in this article.

[$\aleph_0$] "Computer Recreations by 'Aleph-Null', Space-filling Curves, or How to Waste Time With a Plotter", *Software-Practice and Experience*, vol. 1, 403-410, 1971.

[FvD82] James D. Foley and Andries Van Dam. *Fundamentals of Interactive Computer Graphics*, Addison-Wesley, Reading, Mass.,1982. The ordered dither is discussed on pp. 600-602.

[Gar89] Martin Gardner. *Penrose Tiles to Trapdoor Ciphers*, W. H. Freeman and Co., New York, 1989. Includes reprints of articles on fractals and Penrose tiles.

[Gla90] Andrew Glassner. *Graphics Gems*, Academic Press, 1990

[GS86] Branko Grunbaum and G. C. Shephard. *Tilings and Patterns*, W. H. Freeman and Co., New York, 1986.

[Man82] B.B. Mandelbrot. *The Fractal Geometry of Nature*, W. H. Freeman and Co., New York, 1982. Mandelbrot's brief discussion of the Hilbert curve is on page 66.

[Moo00] E. H. Moore. "On certain crinkly curves", *Trans. of the American Mathematical Society*, 1,72-90, 1900.

[OO91a] Yoshio Ohno and Koichi Ohyama. "A Catalog of Symmetric Self-Similar Space-Filling Curves", *Journal of Recreational Mathematics*, 23:3, pp. 161-174, 1991

[OO91b] Yoshio Ohno and Koichi Ohyama. "A Catalog of Non-Symmetric Self-Similar Space-Filling Curves", *Journal of Recreational Mathematics*, 23:4, pp. 247-254, 1991

[OO92] Yoshio Ohno and Koichi Ohyama. "Self-Similar Space-Filling Curves Consisting of Non-Identical Segments", *Journal of Recreational Mathematics*, 24:1, pp. 38-43, 1992. These three articles present many self-avoiding Peano curves that can be produced with von Koch's method.

[Ste86] Paul Joseph Steinhart. "Quasicrystals", *American Scientist*, November/December, 1986, pp. 586-597.

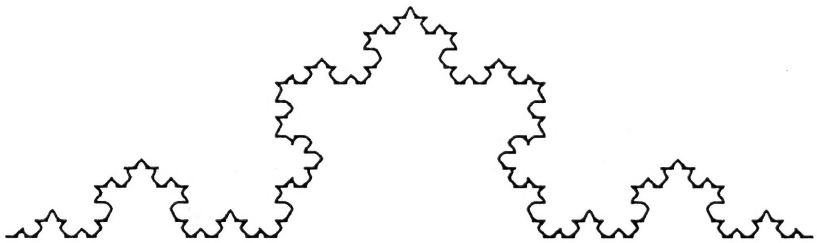

Figure 1. Level 4 Koch Snowflake

## Listing 1

Koch Snowflake

```
  DEFSNG angle,temp,rder,angstep,F
WHILE 1
  CALL MOVETO (2,13)
  INPUT "Level <1-5>";rder
  CLS
  angstep=3.14159/3
  angle=(3/2)*angstep
  F=300/3^rder
  PSET (10,180)
  GOSUB a
WEND

a:
  IF rder>0 THEN
    rder = rder-1
      GOSUB a
      angle=angle+angstep
      GOSUB a
      angle=angle-2*angstep
      GOSUB a
      angle=angle+angstep
      GOSUB a
    rder=rder+1
  ELSE
    LINE -STEP(F*SIN(angle),F*COS(angle))
  END IF
RETURN
```

## Listing 2

Alternate Koch Snowflake

```
  DEFSNG angle,temp,rder,angstep,F
WHILE 1
  CALL MOVETO (2,13)
  INPUT "Level <1-5>";rder
  CLS
  angstep=3.14159/3
  angle=(3/2)*angstep
  F=300/3^rder
  PSET (10,180)
  GOSUB a
  LINE -STEP(F*SIN(angle),F*COS(angle))
WEND

a:
  IF rder>0 THEN
    rder = rder-1
      GOSUB a
      LINE -STEP(F*SIN(angle),F*COS(angle))
      angle=angle+angstep
      GOSUB a
      LINE -STEP(F*SIN(angle),F*COS(angle))
      angle=angle-2*angstep
      GOSUB a
      LINE -STEP(F*SIN(angle),F*COS(angle))
      angle=angle+angstep
      GOSUB a
    rder=rder+1
  END IF
RETURN
```

## Listing 3

Peano Curve

```
  DEFINT dx,dy,temp,rder
WHILE 1
  MOVETO 2,13
  INPUT "ORDER <1-5>";rder
  dx=INT(243/3^rder)
  dy=dx
  CLS
  PSET (30,30)
  GOSUB apeano
  LINE -STEP(dx,dy)
WEND
```

```
apeano:
  IF rder<>0 THEN
    rder = rder-1
      GOSUB apeano
      LINE -STEP(dx,dy)
      temp=dy:dy=dx:dx=-temp'R
      GOSUB apeano
      LINE -STEP(dx,dy)
      temp=dy:dy=-dx:dx=temp'L
      GOSUB apeano
      LINE -STEP(dx,dy)
      temp=dy:dy=-dx:dx=temp'L
      GOSUB apeano
      LINE -STEP(dx,dy)
      temp=dy:dy=-dx:dx=temp'L
      GOSUB apeano
    rder=rder+1
  END IF
RETURN
```

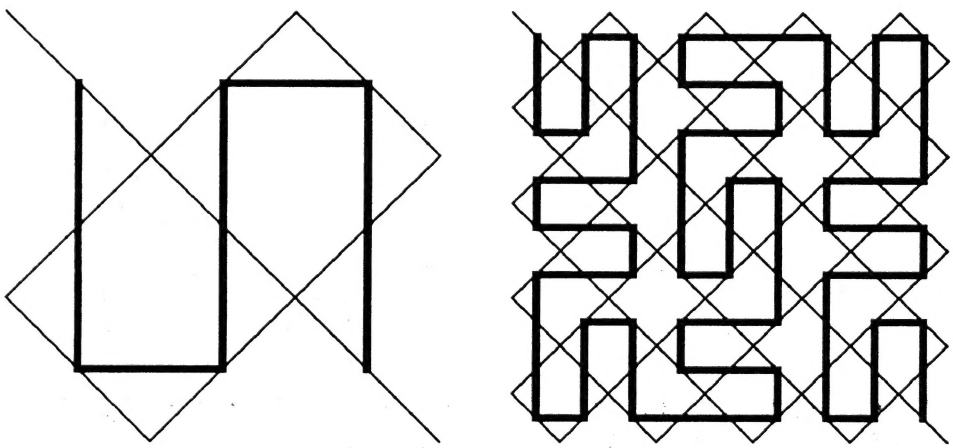

Figure 2. Level 1 and 2 Peano curves (thin lines), and the curves obtained by connecting the midpoints of adjacent segments (heavy lines).

## Listing 4

Open Peano Curve

```
  DEFINT dx,dy,temp,rder
WHILE 1
  MOVETO 2,13
  INPUT "ORDER <1-5>";rder
  dx=0
  dy=INT(243/3^rder)
  CLS
  PSET (30,30)
  GOSUB apeano
WEND
apeano:
  IF rder<>0 THEN
    rder = rder-1
      GOSUB apeano
      LINE -STEP(dx,dy)
      temp=dy:dy=dx:dx=-temp'R
      GOSUB apeano
      temp=dy:dy=-dx:dx=temp'L
      LINE -STEP(dx,dy)
      GOSUB apeano
      temp=dy:dy=-dx:dx=temp'L
      LINE -STEP(dx,dy)
```

```
      GOSUB apeano
      temp=dy:dy=-dx:dx=temp'L
      LINE -STEP(dx,dy)
      GOSUB apeano
      LINE -STEP(dx,dy)
      temp=dy:dy=dx:dx=-temp'R
      GOSUB apeano
      LINE -STEP(dx,dy)
      temp=dy:dy=dx:dx=-temp'R
      GOSUB apeano
      LINE -STEP(dx,dy)
      temp=dy:dy=dx:dx=-temp'R
      GOSUB apeano
      temp=dy:dy=-dx:dx=temp'L
      LINE -STEP(dx,dy)
      GOSUB apeano
    rder=rder+1
  END IF
RETURN
```

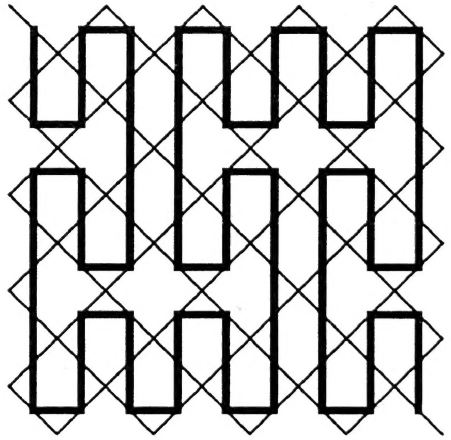

Figure 3. An interpretation of the Peano curve by "Aleph-Null" citing Moore. The original curve (thin lines) appears the same, but the secondary curve (heavy lines) shows that the segments follow a different path.

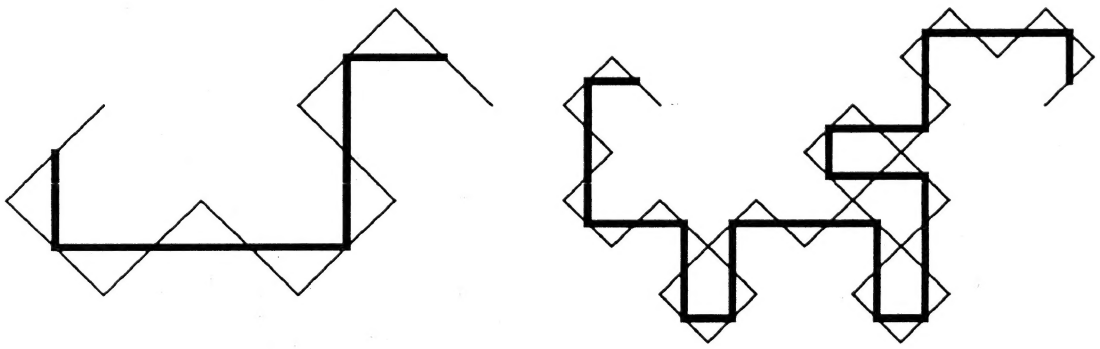

Figure 4. Level 3 and 5 Dragon curves with corresponding "Space dragons".

## Listing 5

Harter-Heighway Dragon

```
  DEFINT dx,dy,temp,rder
WHILE 1
  MOVETO 2,13
  INPUT "ORDER <1-11>";rder
  CLS
  dx=128:dy=0
  FOR i=1 TO rder
    temp=dx:dx=(dx-dy)/2:dy=(temp+dy)/2
  NEXT
  PSET (80,80)
  GOSUB ldragon
  LINE -STEP (dx,dy)
WEND

ldragon:
  IF rder>0 THEN
  rder = rder-1
      GOSUB ldragon
      LINE -STEP(dx,dy)
      temp=dy:dy=-dx:dx=temp'L
      GOSUB rdragon
  rder=rder+1
  END IF
RETURN

rdragon:
  IF rder>0 THEN
    rder = rder-1
      GOSUB ldragon
```

```
      LINE -STEP(dx,dy)
      temp=dy:dy=dx:dx=-temp'R
      GOSUB rdragon
    rder=rder+1
  END IF
RETURN
```

## Listing 6

Space Dragon

```
  DEFINT dx,dy,temp,rder
WHILE 1
  MOVETO 2,13
  INPUT "ORDER <1-11>";rder
  CLS
  dx=64:dy=-64
  FOR i=1 TO rder
    temp=dx:dx=(dx-dy)/2:dy=(temp+dy)/2
  NEXT
  PSET (80,80)
  GOSUB ldragon
WEND

ldragon:
  IF rder>0 THEN
    rder = rder-1
      GOSUB ldragon
      LINE -STEP(dx,dy)
      temp=dy:dy=-dx:dx=temp'L
      GOSUB rdragon
    rder=rder+1
  END IF
RETURN

rdragon:
  IF rder>0 THEN
    rder = rder-1
      GOSUB ldragon
      temp=dy:dy=dx:dx=-temp'R
      LINE -STEP(dx,dy)
      GOSUB rdragon
    rder=rder+1
  END IF
RETURN
```

## Listing 7

Bayer Ordered Dither

```
  defsng d,i,j,k,lim
  CALL TEXTSIZE (10)
WHILE 1
  INPUT"Order<0-4>";lim
  lim=2^(lim-1)
  k=.5
  d=0
  i=0
  j=0
  CLS
  GOSUB setpoint
  IF k<lim THEN GOSUB dither
WEND

dither:
  IF k<lim THEN
    k=k+k
      GOSUB dither
      i=i+k:j=j+k:GOSUB setpoint
      GOSUB dither
      i=i-k:GOSUB setpoint
      GOSUB dither
      i=i+k:j=j-k:GOSUB setpoint
      GOSUB dither
      i=i-k
    k=k/2
  END IF
RETURN

setpoint:
  LOCATE i+1,j*4+1
  PRINT USING "###";d
  d=d+1
RETURN
```

## Listing 8

Penrose Tiles

```
 DEFDBL angle,angstep,long,short,x,y,phi
  DEFINT rder
WHILE 1
  CALL MOVETO (2,13)
  INPUT "Input Order"; rder
  CLS
  angstep=3.14159265#/5
  angle=0
  phi=(1+SQR(5))/2
  short=450/phi^rder
  long=short*phi
  x=450:y=10
```

```
  CALL MOVETO (x,y)
  GOSUB a
WEND

a:
  rder = rder-1
  IF rder=0 THEN
    x=x+short*SIN(angle)
    y=y+short*COS(angle)
    CALL LINETO(x,y)
  ELSE
    x=x+short*phi^rder*SIN(angle)
    y=y+short*phi^rder*COS(angle)
    CALL MOVETO(x,y)
    angle=angle-3*angstep
    GOSUB a
    angle=angle-1*angstep
    GOSUB c
    angle=angle-3*angstep
    GOSUB b
    angle=angle-3*angstep
    x=x+short*phi^rder*SIN(angle)
    y=y+short*phi^rder*COS(angle)
    CALL MOVETO(x,y)
  END IF
  rder=rder+1
RETURN

b:
  rder = rder-1
  IF rder=0 THEN
    x=x+long*SIN(angle)
    y=y+long*COS(angle)
    CALL LINETO(x,y)
  ELSE
    angle=angle-2*angstep
    x=x+short*phi^rder*SIN(angle)
    y=y+short*phi^rder*COS(angle)
    CALL MOVETO(x,y)
    angle=angle-5*angstep
    GOSUB b
    angle=angle-3*angstep
    x=x+short*phi^rder*SIN(angle)
    y=y+short*phi^rder*COS(angle)
    CALL MOVETO(x,y)
    angle=angle-3*angstep
    GOSUB a
    angle=angle+4*angstep
    GOSUB d
    angle=angle-1*angstep
  END IF
  rder=rder+1
RETURN

c:
  rder = rder-1
  IF rder=0 THEN
    x=x+short*SIN(angle)
    y=y+short*COS(angle)
    CALL LINETO(x,y)
  ELSE
    x=x+short*phi^rder*SIN(angle)
    y=y+short*phi^rder*COS(angle)
    CALL MOVETO(x,y)
    angle=angle+4*angstep
    GOSUB c
    angle=angle-3*angstep
    GOSUB b
    angle=angle-4*angstep
    x=x+short*phi^rder*SIN(angle)
    y=y+short*phi^rder*COS(angle)
```

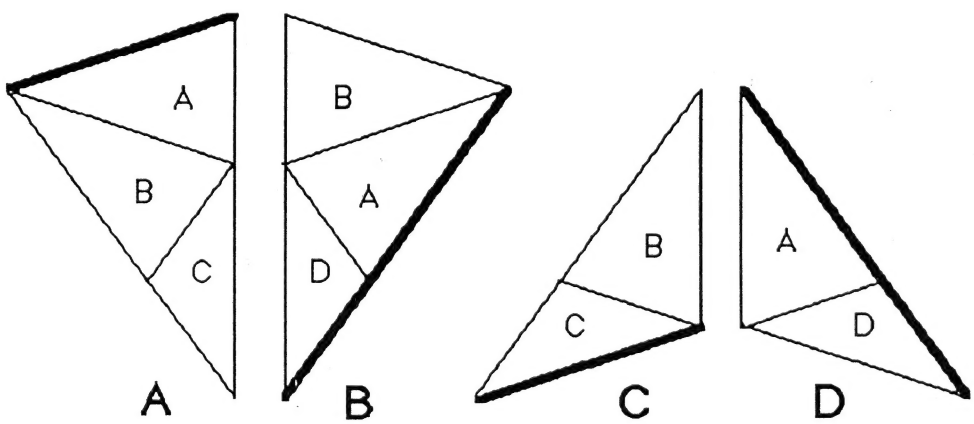


Figure 5. Deflation diagram for Penrose tiles.

```
    CALL MOVETO(x,y)
    angle=angle+3*angstep
  END IF
  rder=rder+1
RETURN

d:
  rder = rder-1
  IF rder=0 THEN
    x=x+long*SIN(angle)
    y=y+long*COS(angle)
    CALL LINETO(x,y)
  ELSE
    x=x+short*phi^(rder-1)*SIN(angle)
    y=y+short*phi^(rder-1)*COS(angle)
    CALL MOVETO(x,y)
    angle=angle+2*angstep
    GOSUB a
    angle=angle+4*angstep
    GOSUB d
    angle=angle+4*angstep
    x=x+long*phi^rder*SIN(angle)
    y=y+long*phi^rder*COS(angle)
    CALL MOVETO(x,y)
  END IF
  rder=rder+1
RETURN
```

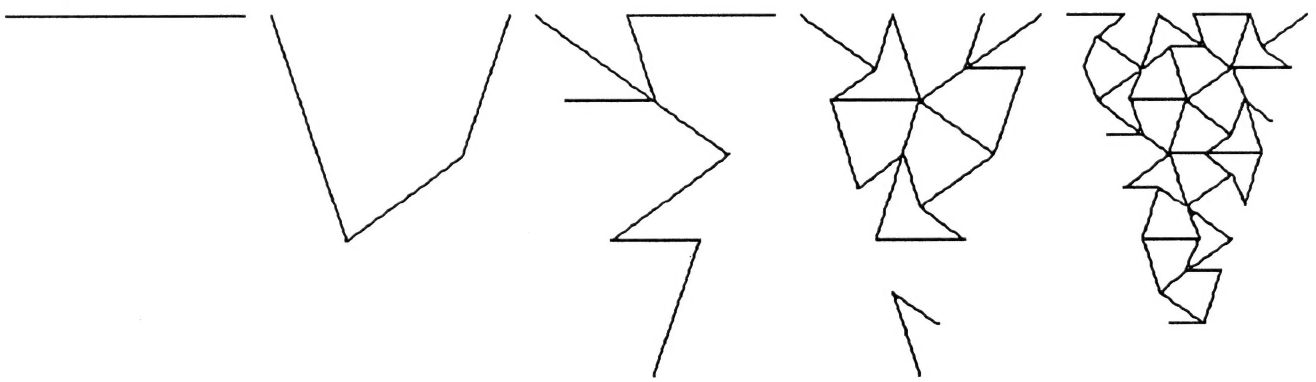

Figure 6. Penrose tiles level 1-5. Each was produced by recursive substitution into diagram "A" in figure 5.

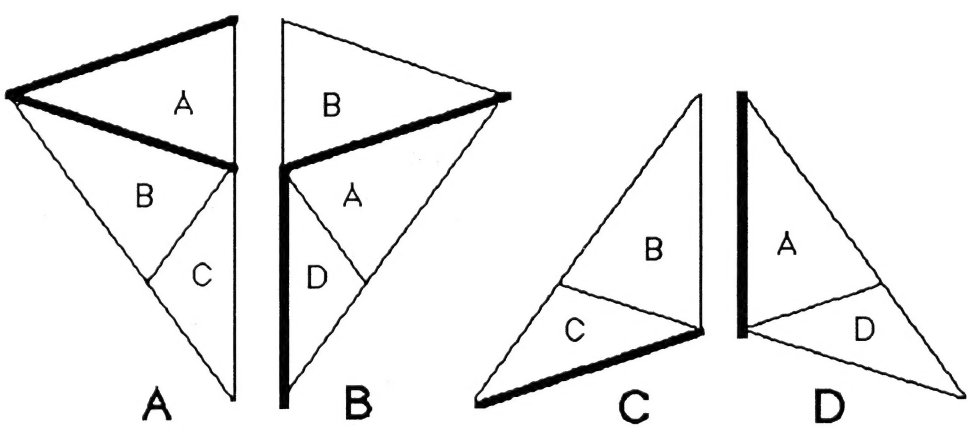


Figure 7.Deflation diagram for rhombus-shaped Penrose tiles. This is identical to figure 5 except that different segments are drawn.

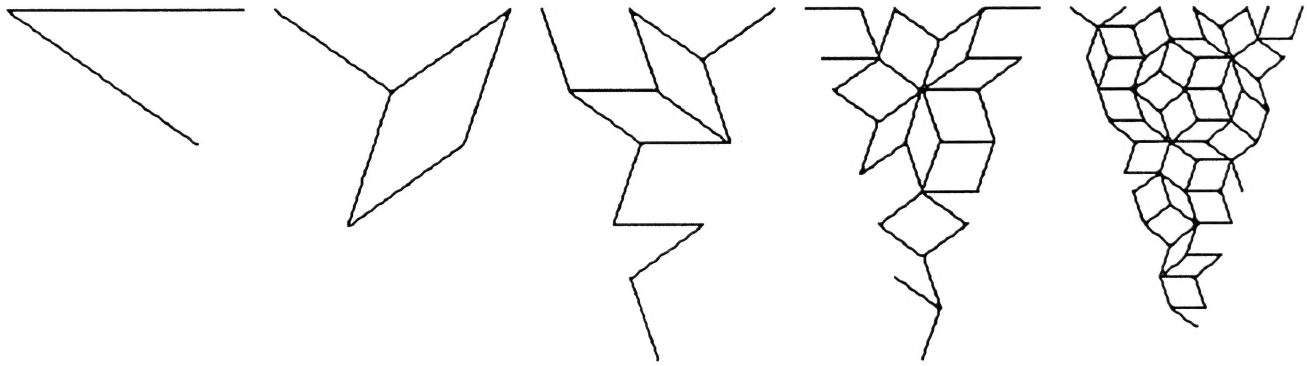

Figure 8. Rhombus-shaped Penrose tiles levels 1-5.

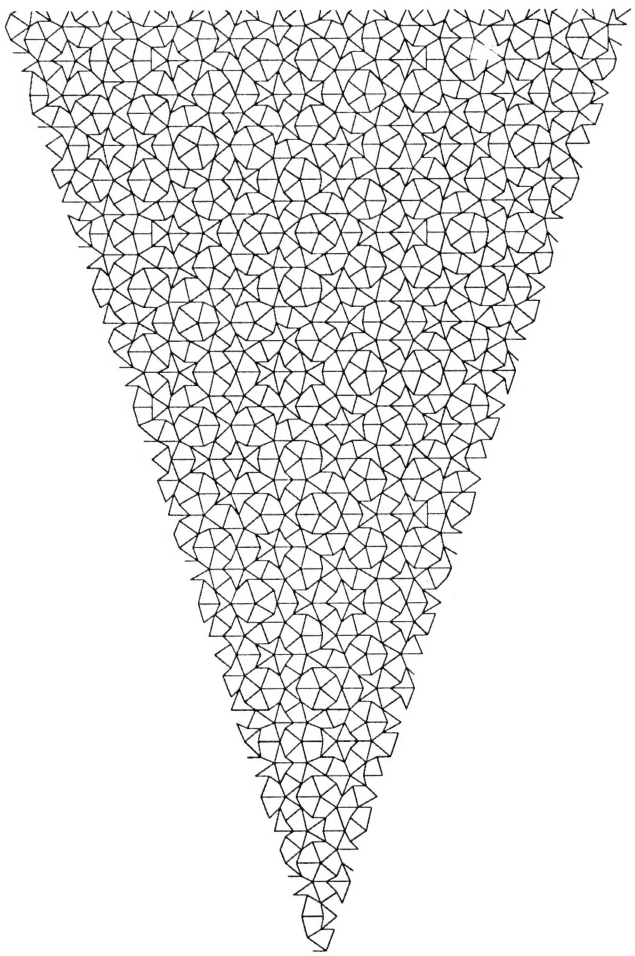

Figure 9. Level 9 Penrose tiles.

# Book Reviews

## Chaos and Fractals

**New Frontiers of Science**

*— H.-O. Peitgen, H. Jürgens and D. Saupe*

1992/984 pages, 686 illustrations, 40 in color

*From the publisher's blurb:*

The fourteen chapters of this book cover the central ideas and concepts of chaos and fractals. They show how these topics relate to each other as well as to many other areas of mathematics and natural phenomena. The book can be understood by readers at almost any mathematical level and is supplemented in each chapter by sections that contain rigorous mathematics for those who want a more in-depth explanation. A "Program of the Chapter" provides computer code for those who wish to explore the concepts graphically. Twenty-four color pages, with forty stunning images, go beyond mathematics to demonstrate the inherent visual beauty of fractals. Two appendices provide the most recent advances in research, the first discusses the details and ideas of fractal image compression and the second explains the foundations and implications of multifractals.

For almost ten years chaos and fractals have shed light on many areas of mathematics and the natural sciences. They continue to do so with power and creativity. This book puts it all at your fingertips.

This book is available from Amygdala for $49.00 plus postage & handling (5# shipping weight): $2.77 (U.S); $9.20 (Canada); elsewhere: consult your postal service.

## Chaos, Fractals and Dynamics

**Computer Experiments in Mathematics**

*Robert L. Devaney*

This book introduces the reader to a popular field in contemporary mathematics: the iteration of functions, and to the study of the results of such iteration: the fascinating images of Julia sets, the mandelbrot set, and fractals. This book is written in a textbook format. It is carefully organized to increase in complexity chapter by chapter, with each chapter building on the results of the exercises, experiments, and programs of previous chapters. It is this well-planned self-teaching organization that makes the book an outstanding self-study guide. As with any mathematics book, it should be read with a pencil in hand to work out the numerous exercises in a work book. In addition, a computer — any computer capable of handling a Basic program — and a rudimentary knowledge of Basic programming are sufficient to run and to modify the programs presented in the text.

The first four chapters review essential algebraic and graphic concepts, and introduce the terminology and concepts of iteration, applied to real numbers. The fifth chapter extends these iteration basics to the complex number domain, and introduces the Julia set. The next two chapters describe two Julia set programs. Their exercises, experiments, and projects provide a wealth of ideas for the study of Julia sets. The eighth chapter is an introduction to the Mandelbrot set, in which the reader is encouraged to write three variations of the computer program supplied. Chapter nine, which deals with geometric iterations, the field of fractals, can be studied independently of the other chapters. The last two brief chapters discuss chaos resulting from the iteration of relatively simple functions, and introduce Julia sets for higher order polynomial functions and transcendental functions, some of which are illustrated on the colored plates at the beginning of the book.

The text material is supplemented with over 50 exercises, usually routine calculations based on text material. There are over 70 experiments that encourage the application of text computer programs and reader-modified computer programs. The text includes nine computer programs written in Basic, ranging in length from seven to 23 lines. The author guides the reader through over 25 projects, each of which results in a specified modification of a text program, or of a previous project program. All programs developed should be saved, so that the reader thereby accumulates over 30 specific programs. In the final chapters of the book frequent references are made to these stored programs, since they are used in advanced exercises and experiments. There are 36 plates with excellent Julia set and Mandelbrot set images in full color, and over 70 figures in the text to inspire the reader's interest and understanding of these results of iterative mathematics.

In any field of mathematics, a real and lasting understanding of the material comes not only from a clear exposition and illustration of the subject, but also from the learner's application of the material under organized guidance. Professor Devaney's expertise supplies these requisites in an interesting and concise text.

— *Review by Richard Scheib*

## Image Lab

*Tim Wegner*

450 page soft glossy back. All about graphics for PC. Comes up to discussion of graphics PD software. Big chapter on Fractint and efforts to give 3D fractals as well as ray tracing, rendering, painting, drawing, fractalising.

Waite Group (publisher)

— *Review by Ian Entwistle*

## Fractal Image Compression

*Michael Barnsley and Lyman Hurd*

This book, a sequel to *Fractals Everywhere,* describes the mathematics underlying the Fractal Transform an image compression method discovered by Michael Barnsley and developed at Iterated Systems. Topics include:

- Iterated Function Systems
- Local IFS and the Black and White Fractal Transform
- The Grayscale Fractal Transform
- Information Theory

- Huffman codes
- Arithmetic Codes
- JPEG Image Compression

Example programs in C are interspersed throughout the text.

## Evolutionary Art and Computers

*Stephen Todd & William Latham (Both IBM, UK).*

Lovely colour illustrations. All about 30 sculpturing big selections by appeal to control mutating evolution. Uses fractal methods.
— *Review by Ian Entwistle*

## Fractals for the Classroom, Part II

*Peitgen, et. al.*

This follows on from Part I, which appeared in 1991. Contains definitive teaching on Mandelbrot and Julia sets.
— *Review by Ian Entwistle*

## Fractal Geometry and Computer Graphics

*J.L.Encarnacio, H-O. Peitgen, G. Sakas, G Englert (Eds.)*

This is a collection of papers from an international workshop held in Darmstadt in the Computer Graphics Center.
Topics vary from fundamentals and new theoretical results to various applications and system developments. Some lovely pics of 30 fractals.
— *Review by Ian Entwistle*

*Amygdala* is published by Rollo Silver, hopefully every four to eight weeks.

Address letters, comments, subscription orders, to: *Amygdala; Box 219; San Cristobal, NM 87564; USA.*

Email: rsilver@lanl.gov Phone: 505/586-0197

Compuserve: Rollo Silver 71174,1453

This issue was produced using FrameMaker on a Macintosh IIcx computer.

## Book Reviews Wanted

If you run across a book on the subject of fractals, etc. that seems particularly illuminating, and has not yet been reviewed in Amygdala, think about writing a review of the book and sending it in for possible publication in Amygdala.

## Submitting Articles

Here are some guidelines for submitting articles for publication in Amygdala.

1. Type your article as you would like to see it appear in the newsletter. Please do not send handwritten drafts!

Send it (*my* order of preference): (A) On 3.5" diskette in Macintosh format for FrameMaker, MacWrite, WriteNow, or MS Word; or as a text file. Please also enclose paper copy so I can see your intent. (B) On 3.5" diskette in IBM format, text file. I have no way to deal with 5.25" diskettes. (C) Paper copy.

2. Illustrations should be either greyscale (suitable for halftoning) or black/white; not color! (A) Normally, illustrations will be printed in full column width, so you should make them 3.25" wide, if possible — provided that they're 300 dpi resolution. If they're grainier, make them larger if possible, so that they'll look good when reduced to 300 dpi. (B) Make sure that you clearly indicate which illustrations go where in the text! (C) All in all, it's better not to have captions welded into your pictures. Let me put them in ad lib. (D) I can handle illustrations on diskette in MacPaint, MacDraw, CricketDraw, Photoshop, Canvas, or Adobe Illustrator formats.

3. Please send along a short biographical note, which I will try to publish in the same issue as your article.

4. Please include your telephone number, in case I have to reach you in a hurry with questions.

## Circulation

As of April 15 1993, Amygdala has 543 subscribers, 183 of whom have the supplemental color slide subscription.